TP1 - Instructions conditionnelles et boucle for

I Les bonnes pratiques des TP d'informatique

I.1 Répertoire de travail

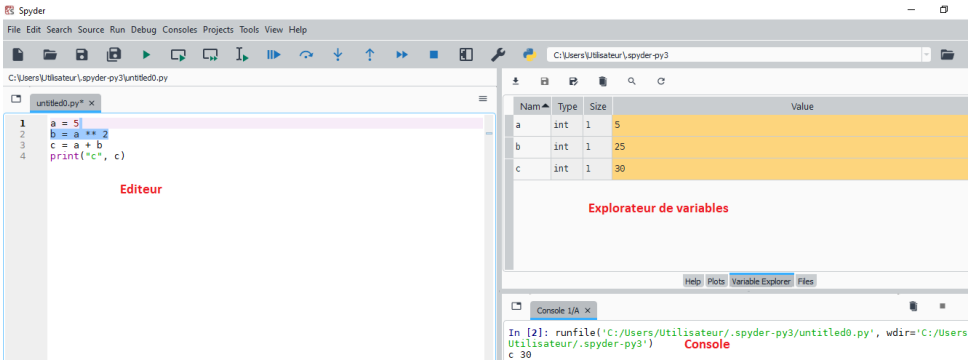
A chaque séance d'informatique :

- Ouvrir l'atelier TP INFO_1A (Ouvrir Console-Elève, puis mes ateliers)
- Copier le dossier du TP
- Coller le dossier du TP dans votre espace de travail `P:\Travail`
- Fermer l'atelier

I.2 Utilisation de `Spyder`

`Spyder` est un environnement de développement (IDE) pour Python qui comprend :

- Un éditeur
- Un explorateur de variables
- Une console
- Etc.



Remarque : A chaque nouveau TP, on ouvre un nouveau fichier qu'on **sauvegarde tout de suite** dans le bon répertoire situé dans l'espace de travail `P:\Travail` avec le nom `TP... .py`

II Les bases de la programmation en `Python`

II.1 Les types de variables

Une **variable** est une zone de la mémoire de l'ordinateur dans laquelle une **valeur** est stockée.
Le programmeur définit cette variable par un **nom**.
En `Python`, on **déclare** et on **initialise** la variable en même temps.

Exemple : On déclare et on initialise la variable `n` à 0.

```
1 n = 0
```

Lors de l'initialisation de la variable, **Python** « devine » le `type` de la variable.

Dans notre exemple, la valeur stockée dans `n` est un entier, donc la variable `n` est de type `int`.

Les principaux types de variables utilisés sont `int` pour les entiers, `float` pour les nombres à virgule flottante, `str` pour les chaînes de caractères et `bool` pour les booléens.

(*Remarque :* un booléen prend 2 valeurs True ou False.)

Exemple :

```
a = 3 # a est de type int
b = 3. # b est de type float
c = '3' # c est de type str
d = (3 == 3) # d est de type bool
```

II.2 Les entrées et sorties

II.2.1 Les entrées

On peut demander à l'utilisateur de rentrer une valeur : on appelle cela une entrée.
On utilise pour cela la fonction `input` qui renvoie une valeur de type `str`.

Exemple :

```
nom = input("Rentrer un nom ")
```

Lorsqu'on exécute cette ligne :

1. le texte `Rentrer un nom` s'affiche dans la console.
2. le programme attend que l'utilisateur rentre un texte (en appuyant sur la touche Entrée).
3. une fois que l'utilisateur a rentré un texte, celui-ci est stocké dans la variable `nom`.

Si on veut que l'utilisateur rentre un entier, on doit convertir en `int` la valeur renvoyée par la fonction `input`.

Exemple :

```
n = int(input("Rentrer un entier ")) # n est de type int
a = float(input("Rentrer un reel compris entre 0 et 1 ")) # a est de type float
```

II.2.2 Les sorties

Pour afficher la valeur d'une variable, on utilise la fonction `print`.

Exemple :

```
a = 5.2
b = 3
print(a)
print(2*b)
```

Sortie
5.2
6

`print(a)` affiche la valeur de la variable `a` *et fait un saut de ligne.*
Remarque : Pour afficher la valeur de plusieurs variables, on les sépare à l'aide d'une virgule.

Exemple :

```
x = "123"
y = 45
z = (3 == 3)
print(x) # affiche 123 puis saut de ligne
print(x, y, z) # affiche 123 45 True puis saut de ligne
print("y =", y) # affiche y = 45 puis saut de ligne
```

Q1. Ecrire un programme qui :

- demande à l'utilisateur de rentrer la longueur d'un carré ;
- affiche le périmètre du carré ;
- affiche l'aire du carré.

Q2. Ecrire un programme qui :

- demande à l'utilisateur de rentrer la base d'un triangle ;
- demande à l'utilisateur de rentrer la hauteur d'un triangle ;
- affiche l'aire du triangle.

II.3 Instructions conditionnelles

`if` signifie « si », `elif` signifie « sinon si » et `else` signifie « sinon ».
`==` permet de tester une égalité
`!=` signifie « est différent de »
`8 % 3` est le reste de la division entière de 8 par 3, soit 2.

Tester les codes suivants :

```
x = 21
if x % 2 == 0:
  print(x, "est pair")
else:
  print(x, "est impair")
```

Sortie

```
x = 21
if x % 3 == 0:
  print(x, "est un multiple de 3")
elif x % 7 == 0:
  print(x, "est un multiple de 7")
```

Sortie

Voici plusieurs syntaxes possibles (la liste n'est pas exhaustive ...)

```
if <condition> :
  <code>
```

```
if <condition> :
  <code1>
else :
  <code2>
```

```
if <condition1> :
  <code1>
elif <condition2> :
  <code2>
elif <condition3> :
  <code3>
```

```
if <condition1> :
  <code1>
elif <condition2> :
  <code2>
elif <condition3> :
  <code3>
else :
  <code4>
```

Q3. Modifier le 2e exemple pour qu'il s'affiche que `x` est un multiple de 3 mais aussi un multiple de 7. Il y a plusieurs solutions.

Q4. Compléter le programme suivant

```
nom = input("Rentrer le nom ")
age = int(input("Rentrer l'age "))
sexe = input("Sexe ? (F/M) ")
# Afficher un message de bienvenue personnalisé suivant le sexe et l'age, par exemple
# Bienvenue Monsieur Durand
# Bienvenue Madame Durand
# Bienvenue Mademoiselle Durand
```

III La boucle `for`

Une boucle permet de réaliser des répétitions.
Il existe deux boucles :

- la boucle `for` : on l'utilise lorsqu'on connaît le nombre de répétitions ;
- la boucle `while` : on l'utilise lorqu'on ne connaît pas le nombre de répétitions.

Exemple :

```
for k in range(3) :
  print(k)
```

Sortie

Commentaires :

- `k` est un nom de variable choisi par le programmeur.
- `for k in range(3)` signifie :
  pour `k` entier allant dans l'intervalle $[0, 3[$ en commençant par 0 et avec un pas de 1.
- Cette instruction permet de coder 3 répétitions.

```
for k in range(1, 4) :
  print(k)
```

Sortie

Commentaires :
`for k in range(1, 4)` signifie :
pour `k` entier allant dans l'intervalle $[1, 4[$ en commençant par 1 et avec un pas de 1.

```
for k in range(10, 4, -3)
    :
  print(k)
```

Sortie

Commentaires :
`for k in range(10, 4, -3)` signifie :
pour `k` entier allant dans l'intervalle $]4, 10]$ en commençant par 10 et avec un pas de -3.

Q5. Ecrire un programme qui affiche 10 fois le texte « Je vais bien ! Tout va bien ! »

Q6. Ecrire un programme qui affiche :
Ligne 1
Ligne 2
...
Ligne 20

Tester plusieurs fois ce programme

```
import random as r
de = r.randint(1, 6)
print(de)
```

Commentaires :

- `import random as r` signifie :
  On importe le module `random` qu'on renomme `r`
  Un **module** est une bibliothèque contenant un certain nombre de constantes, de fonctions.
- A la ligne 2, on utilise la fonction `randint` qui est dans le module `random` renommé `r`
- `random(a, b)` renvoie un entier aléatoire de l'intervalle `[a,b]`.

Q7.

1. Ecrire un programme qui affiche 5 lancers d'un dé.
2. Ecrire un programme qui affiche 5 lancers de 2 dés.

Q8. On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 5$ et pour tout $n \in \mathbb{N}$ $u_{n+1} = 2u_n - n + 10$.
Compléter le programme pour qu'il affiche u_0, u_1, ..., u_5. ($u_5 = 444$)

On rappelle comment on fait cet exercice à la main, pour voir les instructions à coder...
Pour $n = 0$, on calcule $u_1 = 2u_0 - 0 + 10 = \cdots$
Pour $n = 1$, on calcule $u_2 = u_1 + 1 + 10 = \cdots$
...
Pour $n = ...$, on calcule $u_5 = \cdots$
Donc n varie de ... à ...

```
1 # initialisation
2 u = 5
3 print(u)
4 # boucle for
5
6
7
8
```

III.1 Calculer une somme

C'est l'algorithme primordial à connaître. On le retrouve partout ; par exemple, pour calculer une intégrale, on calcule la somme d'aires de rectangles, etc.

Principe

1. On initialise une variable `s` à 0. *C'est dans cette variable qu'on stocke la somme.*
2. On écrit une boucle `for`.
   A chaque itération :
   - on affecte à `s`, l'ancienne valeur de `s` à laquelle on ajoute un terme.
3. On affiche la valeur de `s`.

Exemple :

```
s = 0 # initialisation de la somme
for k in range(4) :
  s = s + k
print(s) # s = 0 + 1 + 2 + 3 = 6
```

Exécution pas à pas du programme

ligne	1	2	3	2						
k		0		1						
s	0		0+0=0							
Affichage										

Q9. Ecrire un programme qui traduit la situation suivante :

- On lance 5 fois un dé.
- A chaque lancer du dé, on affiche le dé.
- A la fin des 5 lancers, on affiche la somme des 5 dés.

III.2 Faire un compteur

Principe

1. On initialise une variable `c` à 0. *C'est dans cette variable qu'on stocke le compteur.*
2. On écrit une boucle `for`.
   A chaque itération :
   - On écrit un test *(qui dépend de ce que l'on souhaite compter.)*
   - Si le test est positif, on incrémente `c` de 1.
3. On affiche la valeur de `c`.

Q10. Ecrire un programme qui traduit la situation suivante :

- On lance 10 fois un dé.
- A chaque lancer du dé, on affiche le dé.
- A la fin des 10 lancers, on affiche le nombre de 6 que l'on a obtenus.

III.3 Trouver le maximum

Principe (Trouver le maximum d'une série de valeurs)

1. On initialise une variable `m` à la plus petite valeur possible de la série. *C'est dans cette variable qu'on stocke le maximum.*
2. On écrit une boucle `for`.
   A chaque itération :
   - si la valeur de la série est plus grande que `m`, alors on modifie la valeur de `m`
3. On affiche la valeur de `m`.

Q11. Ecrire un programme qui traduit la situation suivante :
- On lance 4 fois un dé.
- A chaque lancer du dé, on affiche le dé.
- A la fin des 4 lancers, on affiche le maximum des 4 lancers.

IV Exercices en plus

Q12. Ecrire un programme qui affiche la fréquence d'apparitions du 6 lors de 1000 lancers de dés.

Q13. A l'aide d'une boucle `for`, écrire un programme qui affiche $\sum_{k=0}^{10} k^2$.

Q14. Ecrire un programme qui traduit la situation suivante :
- L'ordinateur affiche 10 nombres entiers aléatoires entre 1 et 100.
- L'ordinateur affiche ensuite le plus petit intervalle qui contient tous ces nombres.

Q15. Ecrire le programme qui affiche la table de multiplication d'un entier donné par l'utilisateur. Par exemple, si l'utilisateur veut la table de 9, il s'affiche :
1 x 9 = 9
2 x 9 = 18
...
10 x 9 = 90

Q16. Ecrire un programme qui propose à l'utilisateur 10 sommes à calculer (sans calculatrice). A la fin des 10 réponses, on affiche le score de bonnes réponses et un message donnant des conseils suivant le score obtenu.